CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4F13	Title of report	4F13 Coursework 1

Date submitted: 04/11/2022	Assessment for this module is [✓ 100%] / ☐ 25% coursework of which this assignment forms 33.3 %

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5674E	Name:		College:	

Feedback to the student

☐ **See also comments in the text**

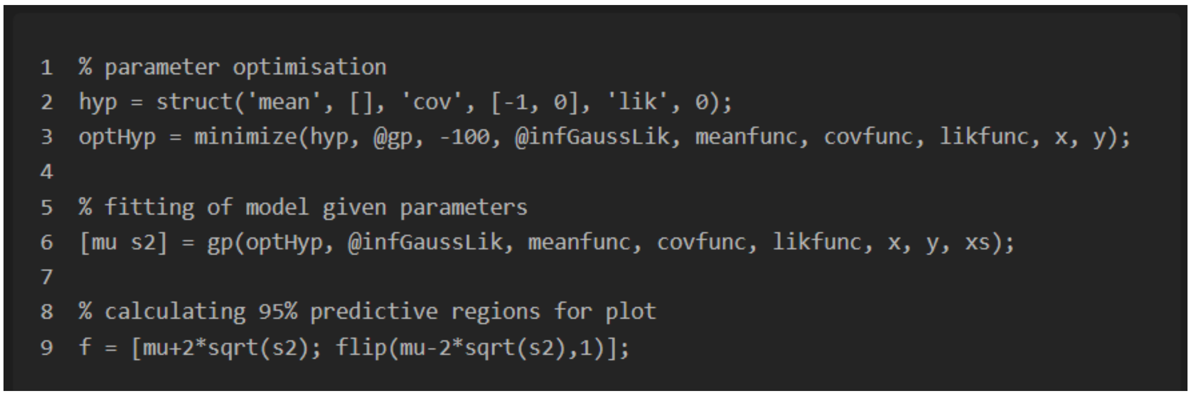
		Very good	**Good**	Needs improvmt
C O N T E N T	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
P R E S E N T A T I O N	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Indicative grades are not provided for the FINAL piece of coursework in a module

Assessment (circle one or two grades)		A*	A	B	C	D
Indicative grade guideline		>75%	65-75%	55-65%	40-55%	<40%
Penalty for lateness:		*20% of maximum achievable marks per week or part week that the work is late.*				

Marker: Date:

Task A



```
1  % parameter optimisation
2  hyp = struct('mean', [], 'cov', [-1, 0], 'lik', 0);
3  optHyp = minimize(hyp, @gp, -100, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
4
5  % fitting of model given parameters
6  [mu s2] = gp(optHyp, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
7
8  % calculating 95% predictive regions for plot
9  f = [mu+2*sqrt(s2); flip(mu-2*sqrt(s2),1)];
```

Figure 1 – Relevant code from task A

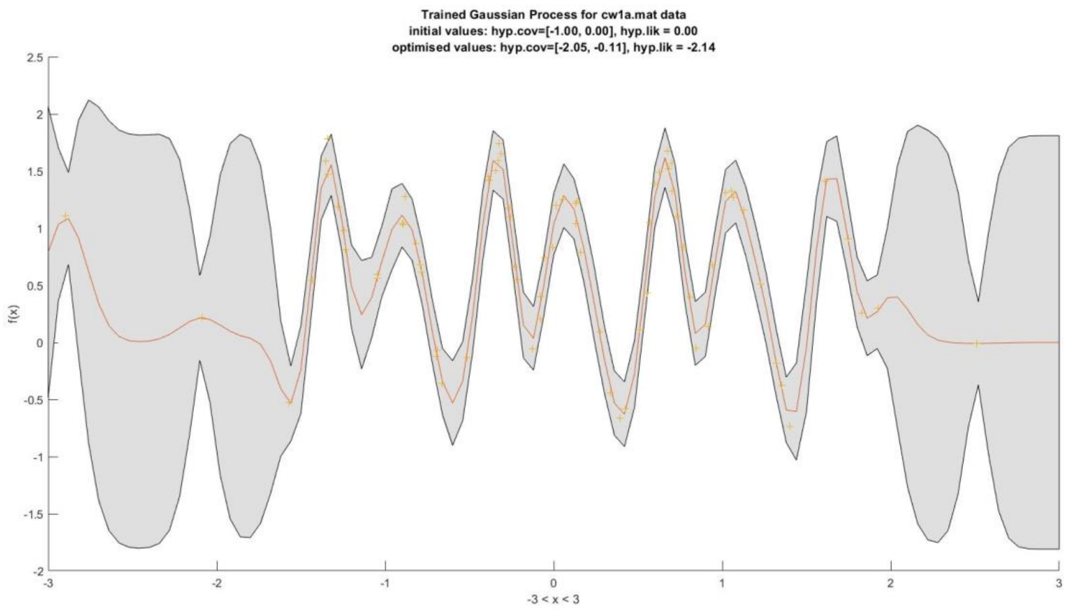
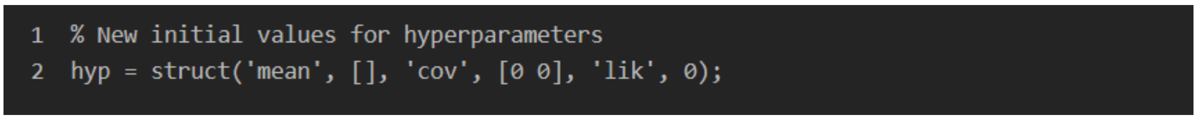


Figure 2 - GP fit to data in cw1a.mat

We optimise model parameters using figure 1 line 1, this maximises the marginal likelihood: the probability of output labels given the model form, independent of parameters. The log marginal likelihood has two terms, one to ensure data fit, and a prior term to penalise model complexity, these ensure Occam's Razor considerations. A local optimum for the parameters are [0.128, 0.897, 0.118]. The first parameter is a length scale for the squared exponential covariance function of the GP. The value given is close to the spacing of the data, thus model output is based on nearby points on the domain leading to the tight fit to the data seen in figure 2. The second parameter is an amplitude parameter determining individual contribution of points to the model, this value is large enough that individual points contribute to the distribution, as can be seen in regions with little data a single point greatly improves confidence, however it is small enough that the model is not unduly influenced by outliers and isn't overconfident in regions with lots of data. The third parameter is model noise, a prediction on how noisy the generation of the data from the 'true' model was. The optimised noise is small enough that the data can be fit tightly as in figure 2, but there is still uncertainty around the mean function, again ensuring the model doesn't overfit.

Task B



```
% New initial values for hyperparameters
hyp = struct('mean', [], 'cov', [0 0], 'lik', 0);
```

Figure 3 – Relevant code from task B

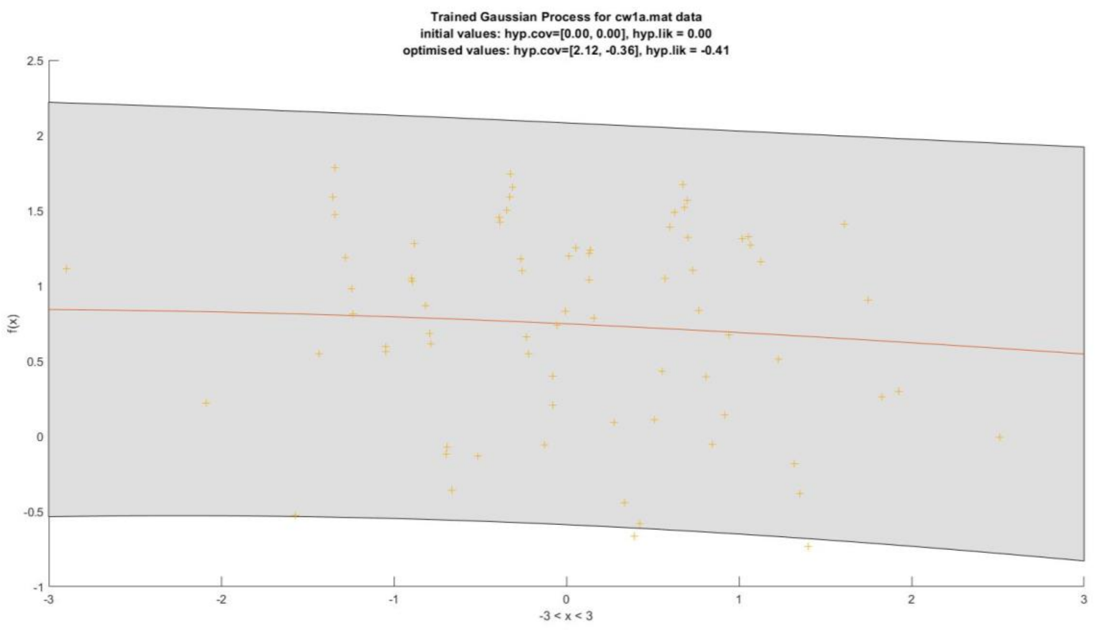
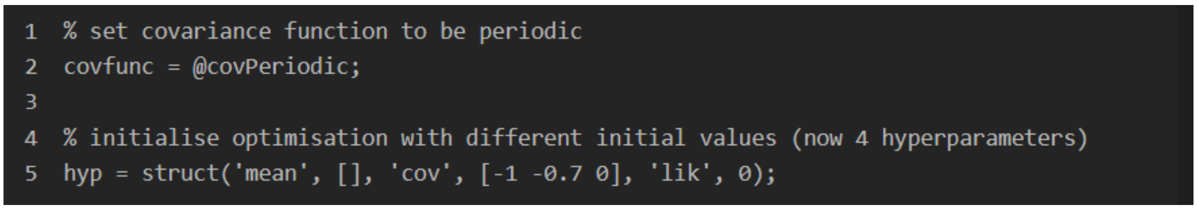


Figure 4 - GP fit to data in cw1a.mat

A different local maximum of the marginal likelihood is [8.35, 0.70, 0.66]. The length parameter is now much larger, a much wider range of the data influences the model output at each point in the domain, this leads to the model averaging over all of the data. The amplitude parameter has not changed much, but it is less prominent here because the length term in the covariance is larger, so the K matrix is less sparse, individual point contribution is therefore less prominent because an evaluation of the predictive distribution at a particular point is influenced by more of the data. The noise term is larger, contributing to the much less confident fit, however again because the values in K are larger the noise is less prominent in the predictive distribution. The value of this marginal likelihood optima is lower than that from task A, so independently of parameters, the model from task A is more likely. We can see this clearly from the fit, it seems unlikely that the predictive distribution in figure 3 generated the data shown, there are very clearly more intricacies to the distribution, the model has under-fitted here.

Task C



```
% set covariance function to be periodic
covfunc = @covPeriodic;

% initialise optimisation with different initial values (now 4 hyperparameters)
hyp = struct('mean', [], 'cov', [-1 -0.7 0], 'lik', 0);
```

Figure 5 – Relevant code from Task C

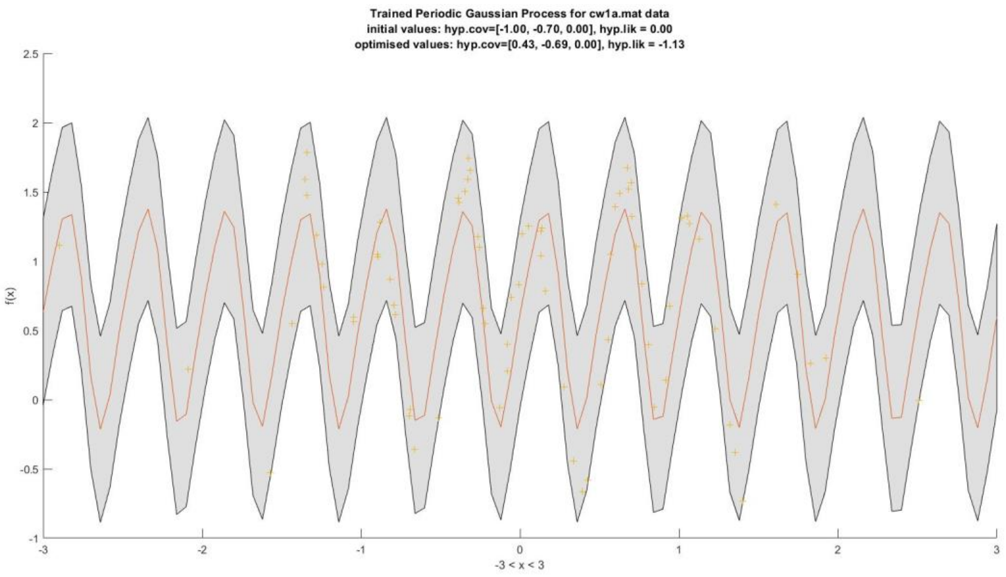
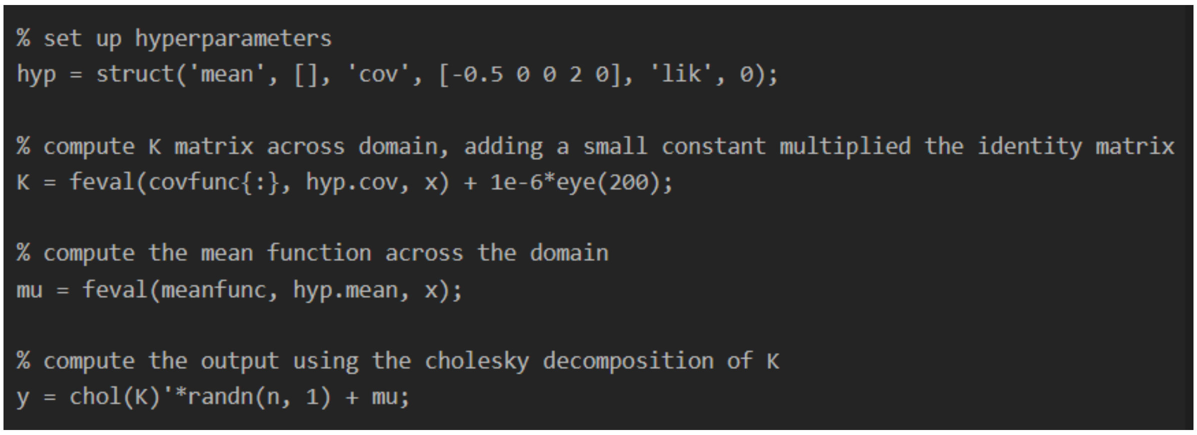


Figure 6 - GP fit to data in cw1a.mat with a periodic covariance function

The covariance function has been altered so that another parameter has been added to restrict the data to represent a point along a period. The maxima of the marginal likelihood in this case is [1.54, 0.547, 0.323, 0.5], where the final parameter is now a period. We can see from figure 5 that the model is generally less confident about the data than in task A, this is because the uncertainty must be periodic and so there is contribution from regions with and without data, leading to a less confident model. However, it is more confident in regions with less data, as this is being influenced by the periodic restriction and the areas with data. It seems an intuitively poor decision to increase the confidence in regions without data and decrease it in the regions with data. The marginal likelihood for this selection of hyperparameters is lower than for the model in task A, implying that a periodic distribution is on the whole a poorer model here, and it is unlikely that the generating function was originally periodic.

Task D



```
% set up hyperparameters
hyp = struct('mean', [], 'cov', [-0.5 0 0 2 0], 'lik', 0);

% compute K matrix across domain, adding a small constant multiplied the identity matrix
K = feval(covfunc{:}, hyp.cov, x) + 1e-6*eye(200);

% compute the mean function across the domain
mu = feval(meanfunc, hyp.mean, x);

% compute the output using the cholesky decomposition of K
y = chol(K)'*randn(n, 1) + mu;
```

Figure 9 – Relevant code for Task D

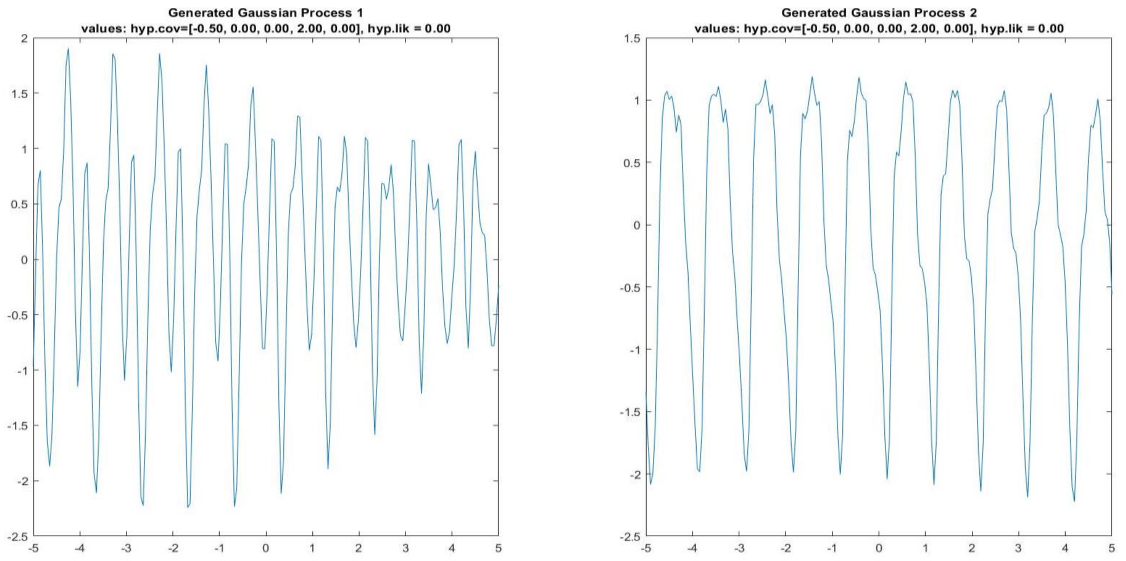
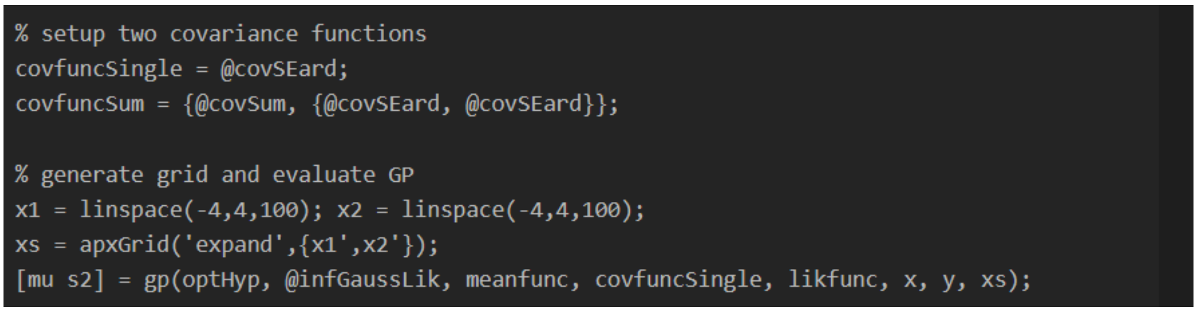


Figure 8 – Gaussian Processes Generated from hyp.cov = [-0.5, 0.0, 0.0, 2.0, 0.0] and hyp.lik = 0.0

We can use a covariance function that is a product of a squared exponential and a periodic function, each with their own hyperparameters, two length scales, two amplitudes and a period. Setting the length scale in the periodic function to 0.61 and the one in the standard squared exponential to 7.4 yields random processes as seen in figure 7. The small length scale in the periodic function leads to the K matrix having large fluctuations between nearby values, that is consistent across periods. The larger length scale in the standard squared exponential leads to slower fluctuations across the average of points as can be seen from the deviations between periods. To generate the process itself we generate a vector of random gaussian variables and multiply by the Cholesky decomposition of K (figure 8), this ensures that we have a random process. However first we must add a small identity matrix to K to ensure it is positive definite and we can compute the Cholesky decomposition (figure 8). This joint generation gives a random process with the properties determined by the form of the covariance function.

Task E



```
% setup two covariance functions
covfuncSingle = @covSEard;
covfuncSum = {@covSum, {@covSEard, @covSEard}};

% generate grid and evaluate GP
x1 = linspace(-4,4,100); x2 = linspace(-4,4,100);
xs = apxGrid('expand',{x1',x2'});
[mu s2] = gp(optHyp, @infGaussLik, meanfunc, covfuncSingle, likfunc, x, y, xs);
```

Figure 9 – Relevant Code for Task E

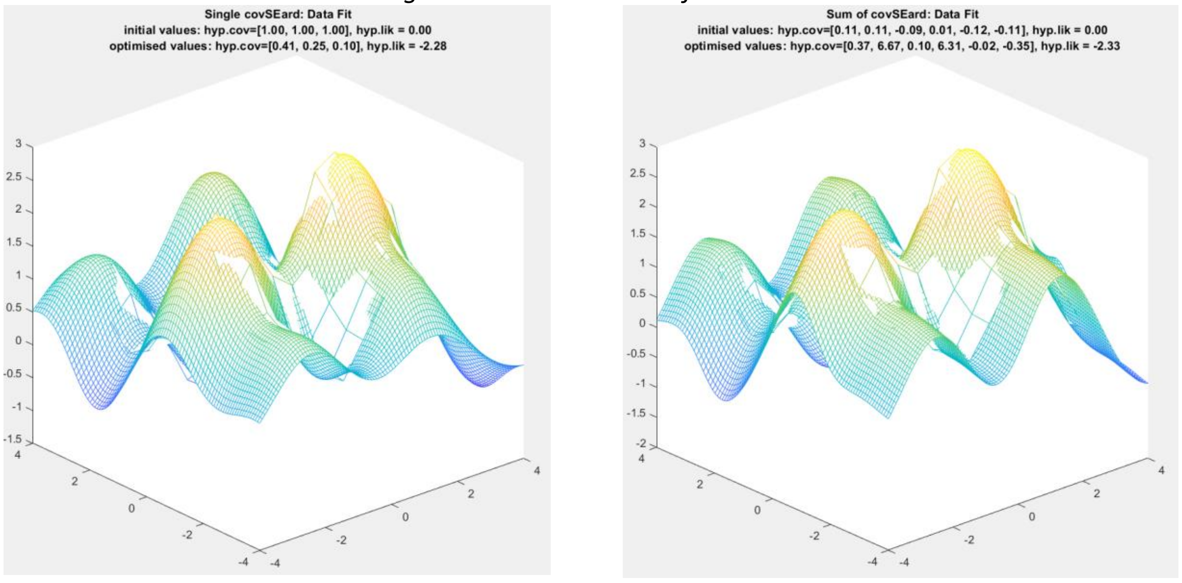


Figure 10 – Fit of mean function to data for single covariance and summed covariance

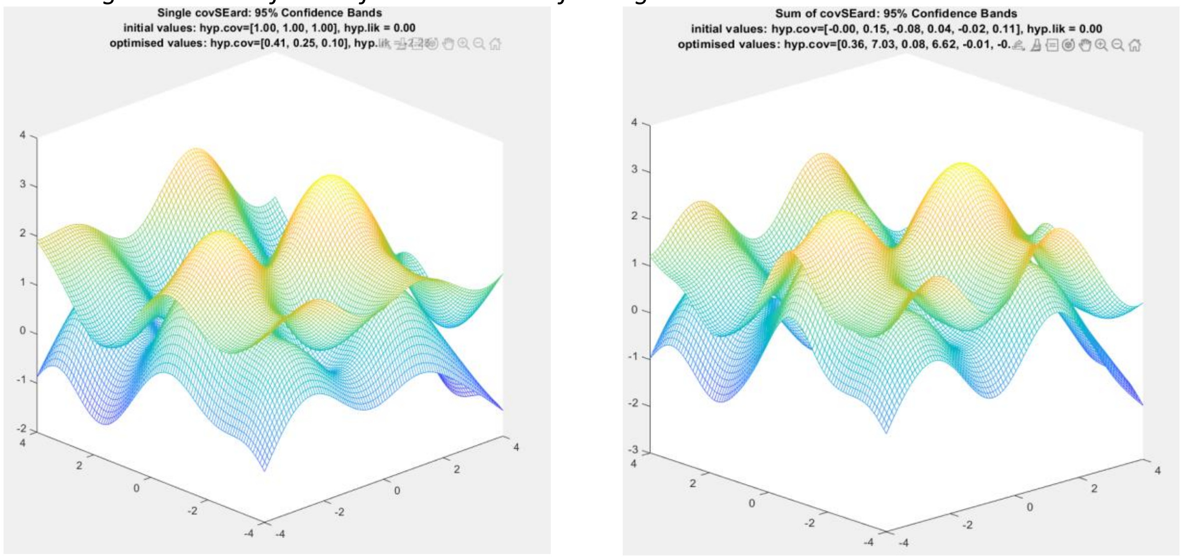


Figure 11 – 95% Confidence regions for data fit for single covariance and summed covariance

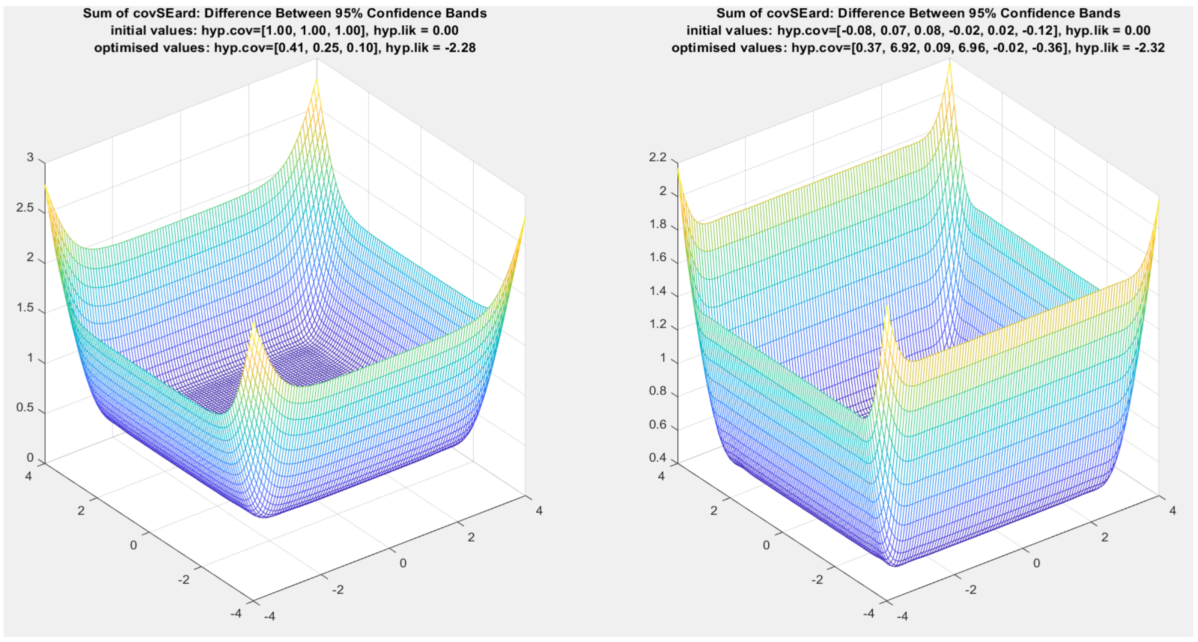


Figure 12 – Difference between upper and lower 95% Confidence regions for data fit for single covariance and summed covariance

We consider two forms of covariance function, the first is a single multi-variate squared exponential, the second is a sum of two multi-variate squared exponentials. A sum of two covariance functions makes K denser, as the output of a squared exponential is always positive, this increases the determinant of K, thus decreasing the marginal likelihood (a complexity penalty), but it also increases the marginal likelihood from the data fit term so long as the added complexity allows the data to be better fit. We see from comparing the marginal likelihoods that the sum of exponential covariance functions improves the marginal likelihood, and so the improved data fit outweighs the increased complexity. Although the mean functions look similar (figure 10), we can see from figures 11 and 12 that the increased complexity of the model reduces uncertainty. This is because for the single SE function, there are two similar length scales along both axes, neither of which can account for larger trends across the domain. Whereas there is anti-symmetry in the summed squared exponential, which allows the two covariance functions to account for small and large length scales along both axes of the input domain.

Total 979 words excluding code and captions